Using Adobe® LiveCycle® Production Print ES in a cluster configuration

Table of contents

1 Introduction
3 Planning the cluster
4 LiveCycle Production Print ES components
7 Database for LiveCycle Production Print ES repositories
8 Postprocessor repository
10 Printers and print servers
11 StreamStudio
15 Testing the cluster

Introduction

This document provides an overview of how to set up Adobe LiveCycle Production Print ES software in a cluster environment. It offers guidelines for achieving a failover solution that load balances incoming jobs.

The communication server (also known as StreamServer) is the core engine of LiveCycle Production Print ES. You use several StreamServers, and place the repositories on a shared resource area. Incoming jobs are processed by an active-active cluster solution (that is, traffic intended for a failed StreamServer is passed on to another available StreamServer).

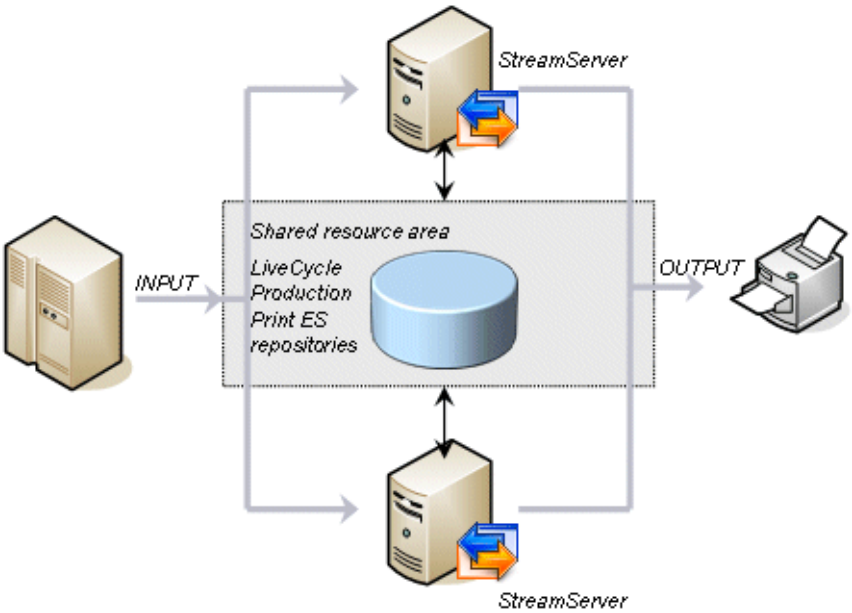

Figure 1. Example of a LiveCycle Production Print ES cluster environment

Powered by StreamServe™

Adobe®

Intended audience

This document is intended for:

- Database administrators
- System administrators
- Developers, for example LiveCycle Production Print ES consultants

It is assumed that the reader has basic knowledge about cluster technologies and is also familiar with LiveCycle Production Print ES.

Information about third-party products

This document does not describe how to carry out configurations in third-party products, for example, Microsoft Cluster Administrator. For such information, see the user documentation for the related product.

Cluster concepts

This section provides a brief overview of the different components of a cluster.

A cluster is monitored and handled by a platform-dependent cluster software. For example:

- For Microsoft® Windows®, you can use Microsoft Cluster Server (MSCS).
- For UNIX®, you can use Veritas Cluster Server (VCS), which supports most UNIX platforms such as Solaris™, Linux®, and AIX® or you can use IBM® High Availability Cluster Multiprocessing (HACMP) for AIX.

Each cluster provides an administration module to be used when configuring the cluster. For example, for MSCS you can use the Cluster Administrator tool. In the administration module, you configure the cluster including cluster nodes, cluster groups, and cluster resources.

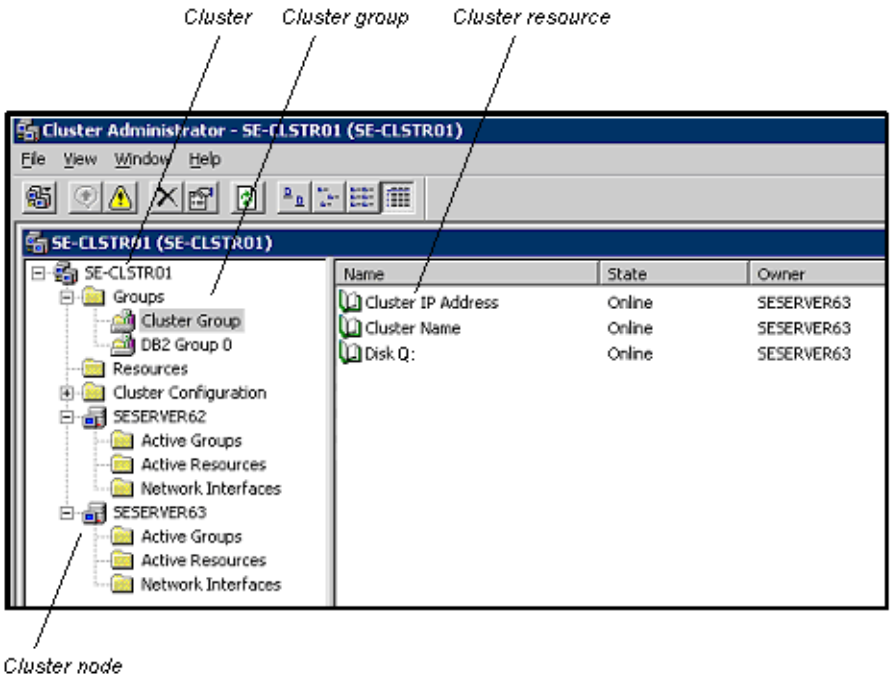

Figure 2. Cluster configuration in the Microsoft Cluster Administrator tool

When you configure a cluster in Microsoft Cluster Administrator, a default cluster group is automatically created, including an IP address, a network name, and a quorum disk. The quorum disk stores a copy of the cluster configuration to help ensure that the cluster configuration always remains consistent at any failure.

You must not install any other resources in this default cluster group.

A node is a physical server in the company network. The most common size for a failover cluster is two nodes, which is the minimum to provide a failover. However, a cluster can consist of many more nodes, sometimes even dozens.

In general, it is recommended to install as much as possible on the cluster node, and only to install common resources, such as file shares and database files and tables, on the shared resource area.

Cluster groups

Apart from the nodes, you must also set up cluster groups where you configure the cluster resources (see "Cluster resources"). Each cluster group requires a separate disk on the shared resource area.

The cluster groups reside on one of the physical servers. At a failure, the cluster groups are moved from one server to another in a matter of seconds.

Possible and preferred owners

Each cluster group is owned by one of the cluster nodes at any given time. You can assign possible owners and preferred owners (nodes) for each cluster group. If you only assign possible owners, the cluster groups are moved once at a failover to another possible owner. When the original node is online and healthy again, the cluster groups remain on the second node. This saves time and resources, especially if large amounts of data are to be moved.

However, in some situations you may want to assign different preferred owners. For example, if you run two different cluster groups in the same cluster, you can configure them to have different preferred owners. They will then charge to different servers, but act as failover for each other.

Cluster resources

Each cluster group includes cluster resources. A resource is an entity managed by the cluster software, for example, an IP address, a host name (network name), a physical disk, and one or several cluster services. A cluster service could be, for example, a generic service or a file share.

Each resource type is equal to a monitor agent type in the cluster software. Different types of monitor agents require different configurations. You can, for example, assign possible and preferred owners and designate dependencies to other already created resources in the same cluster group.

Cluster terminology

The following terms are used in this document:

Scalability (horizontal scaling)
The ability to add new physical servers to a cluster solution in order to increase job throughput. As the servers can be added to the cluster at runtime, zero downtime is required.

Availability
The ability to keep a service online, even if one or more physical servers are taken offline, for example, due to failure, maintenance requirements, and so on.

Load balancing
The ability to distribute a load across several servers.

Failover
The ability to automatically switch over to a redundant or standby server in case of failure.

Fault tolerance
The ability not to lose a job that is being processed at the time a failure occurs.

Planning the cluster

You must plan the cluster and decide what cluster nodes and cluster groups are to be used. To eliminate every potential single point of failure (SPOF), all members of the cluster must allow failover.

How to set up the cluster depends on the company network environment, the expected loads, the performance requirements, and so forth. There may even be an already existing cluster on the network, which LiveCycle Production Print ES will share with other applications.

A general recommendation is to install as much as possible on the cluster nodes and to use only the shared resource area for common parts.

Basic environment

In a basic environment, it may be sufficient to set up a cluster with two nodes (physical servers), and install LiveCycle Production Print ES software (including the postprocessor repository) and the database software on both nodes.

Complex environment

A more complex environment may require several nodes, where StreamServer, the postprocessor repository, and the database software are installed on different nodes. You may even consider using several clusters.

High-volume scenarios

When high volumes are expected, and when two physical StreamServers are not expected to manage the load, you should not install the StreamServers on the same nodes as the database software and the postprocessor repository.

Instead, install the StreamServers on separate nodes. These nodes should not be clustered, but the load should be balanced between the servers via scheduled spooling. For extreme loads, you may even consider using more than two StreamServers, if the necessary hardware is available.

An installation with StreamServers on separate nodes has the advantage of creating a cleaner installation with nothing but clustered applications in the cluster. This approach is also recommended if the database and the postprocessor repository will share the cluster with other clustered applications.

LiveCycle Production Print ES components

This section describes how you set up and use LiveCycle Production Print ES subcomponents, for example, StreamServer and the management gateway, in a cluster.

LiveCycle Production Print ES components (except the postprocessing repository) are not cluster-aware and should not be set up as cluster resources. However, these components do support horizontal scaling. This means that you can install the LiveCycle Production Print ES software locally on the cluster nodes, or you can add new nodes, with LiveCycle Production Print ES installed. Even though the components are not handled by the cluster software, you will still achieve load balancing and failover. When it comes to fault tolerance, LiveCycle Production Print ES can correct for failures that occur within StreamServer, the network, or the runtime repository.

Local and common storages

On each node where the LiveCycle Production Print ES software is installed, you must run identical StreamServer Projects. Since the applications run locally, you must store the configuration files (export files) and working directories locally. The folder structure of the working directories must be identical on all nodes.

In many cases, a common storage area is needed for processing, for instance to hold the input directory (if a directory input connector is used) or the output directory (if a file output connector is used). You also use this area for common files with data that originates from several jobs,

for example, table files with information about job processing. The best place to put this area is on a network file share. To make sure the file share is also fault tolerant, it must be installed as a resource in the cluster. All StreamServer applications will access the directories and the files through the file share.

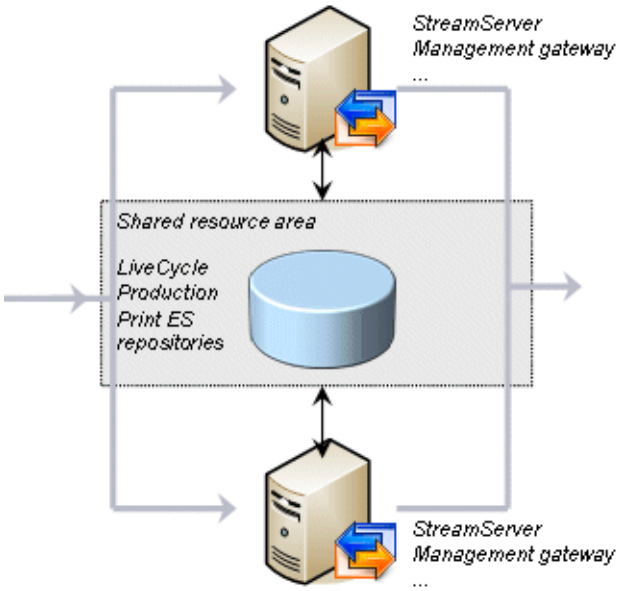

Figure 3. Example of LiveCycle Production Print ES components in a cluster with two nodes

Load balancing and failover

If two or more StreamServers are running identical Projects and are sharing the queues, one of them can take over a job if the currently processing StreamServer fails. If all StreamServers are alive and functioning, the incoming jobs are load balanced among the StreamServers. LiveCycle Production Print ES does not split jobs, and load balancing and failover are achieved per job.

All of the following are required in order to achieve load balancing and failover:

- Load balancing via shared queues and scheduled spooling
- Monitoring via heartbeats
- Failover via heartbeatevents

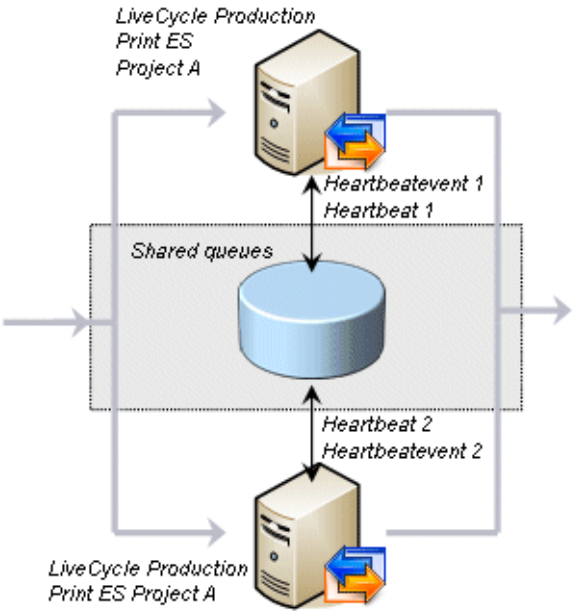

Figure 4. Example of heartbeats and heartbeatevents in a cluster

Load balancing via shared queues and scheduled spooling

If you run identical Projects on all nodes in a cluster, the StreamServers will share the same input and output queues. To enable load balancing, you must select **Schedule spooling** in Design Center. The spooling interval (by default one minute) specifies how often each StreamServer polls a queue for new jobs when it is idle.

Under certain circumstances, you can increase job throughput by decreasing the default spooling interval. For example, in order to enhance response times when processing HTTP calls from web applications, you can decrease the spooling interval.

For information on how to select and schedule the spooling, see the Design Center documentation.

Monitoring via heartbeats

Since StreamServer itself is not monitored by the cluster, no cluster agent will control the included StreamServers. Instead, heartbeats are used to monitor the status and health of the StreamServers. During processing, every 30 seconds (default heartbeat interval) until the job is completed, each StreamServer updates the runtime repository with job-thread status information (whether the job is in processing or completed).

For information on how to edit the heartbeat, see the *Database Guidelines*.

Failover via heartbeatevents

Heartbeatevents enable failover of incoming jobs. Every ten minutes, the StreamServers read the job-thread status information from the runtime repository. If the status of a job thread in processing state has not been updated in the last 30 seconds (heartbeat interval), the job fails over to another available StreamServer.

The default heartbeatevent interval is ten minutes. You can schedule the heartbeatevent in order to rerun uncompleted jobs in a more controlled manner.

Note: Failover is carried out for jobs in processing state (when the job is between the input queue and the output queue). Failed jobs (when the job has not been received by the input connector or not been sent from the output connector) will not fail over. To enable resending of failed jobs, you must specify the number of retries in Design Center. For more information, see the Design Center documentation.

For information on how to edit and schedule the heartbeatevent, see the *Database Guidelines*.

Setting up StreamServer in a cluster

This section includes a few tips and recommendations for setting up the LiveCycle Production Print ES components in a cluster.

- Install LiveCycle Production Print ES software locally on each node. Select a local directory for the working directories for the LiveCycle Production Print ES applications. See the *Installation Guide*.
- Configure the Project in Design Center. For details, see the Design Center documentation.
  - Enable **Schedule spooling**. If required, change and schedule the default spooling interval.
  - Select a local directory for the export files.
  - Export identical Projects to all the nodes where StreamServer resides.
  - Place all common files (with data that originates from several jobs) in directories on the shared resource area.

- Create the application domain and deploy the Projects from Control Center. Use one application domain for the entire environment (however, different application domains for development, test, production, and so on). For details, see the Control Center documentation.
  - When you specify the database to be used for the application domain, assign the network name of the clustered database instance (not the host name of the specific LiveCycle Production Print ES node).
  - Since the LiveCycle Production Print ES applications are not started by the cluster, you must set up the applications to start automatically.
- Set up the required cluster resources in the cluster administration module:
  - A physical disk
  - A file share resource for the shared resource area

    If you intend to include the postprocessing repository (see "Postprocessor repository") in the same cluster group, it is recommended that you create a subdirectory for the file share. For example, if you include a postprocessing repository resource in a cluster group that uses D: as the physical disk, you can create a D:Spoolsub-directory for the file share.

  For example, in MSCS:

LiveCycle Production Print ES Application Group	
Resource name	**Resource type**
Disk D:	Physical disk
File share for the common storage	File share

Database for LiveCycle Production Print ES repositories

The LiveCycle Production Print ES repositories (enterprise, runtime, and web content) reside on a database instance (Microsoft SQL Server, Oracle Database, or IBM DB2 Universal Database). The database instances are fault tolerant and cluster-aware and can be managed by the cluster software.

Database cluster configurations

When clustering a database, you can either use a hot standby or mutual takeover configuration. In a hot standby configuration, at least one database server in the cluster is idle and dedicated as a backup in the event of a failure. In a mutual takeover configuration, all database servers are active and participate in the cluster

Database software and common files

The database software must be installed on the local disk of each node. You can install the database software on the same nodes as the LiveCycle Production Print ES components, or you can use separate nodes. The database files and log files for the LiveCycle Production Print ES repositories must reside on a disk on the shared resource area.

Connections

The database software accesses the shared resource area using ODBC or JDBC.

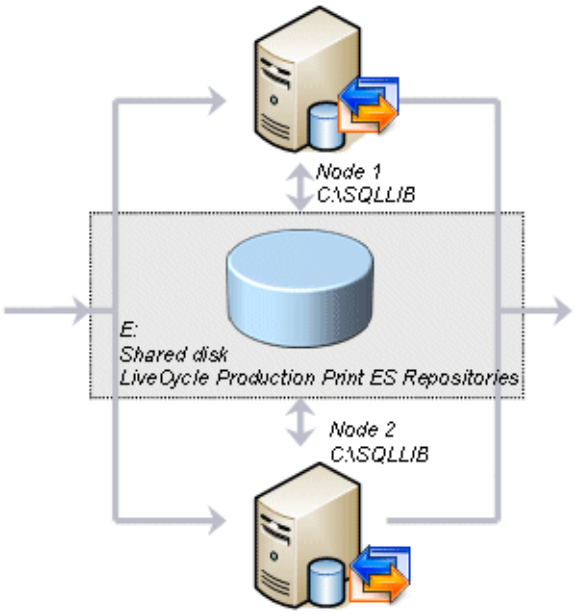

Figure 5. Example of a database cluster with two nodes (Windows)

Setting up a database in a cluster

Clustering databases is a complex task, which is beyond the scope of this document. For detailed information on how to cluster a database, see the user documentation for the relevant database type.

To cluster a database, you must have detailed knowledge about the database. It is therefore recommended that the clustering be performed by a database administrator.

Example of an SQL cluster group with cluster resources

In this example, a mutual takeover cluster is set up in MSCS. The following cluster group and cluster resources are used.

SQL Server Group	
Resource name	**Resource type**
Disk E:	Physical disk
SQL IP address	IP address
SQL network name	Network name
SQLinstance	SQL Server
SQL Server agent	SQL Server agent
SQL Server fulltext	Generic service

Postprocessor repository

If you intend to use the Document Broker component or Communication Reporter, you must install a postprocessor repository in the cluster. The repository runs on a FastObjects database by Poet and is cluster-aware.

You can install the postprocessor repository locally on the same nodes as StreamServer and the database instance, or you can use separate nodes. If you run the repository and the database in the same cluster, it is recommended that you use separate cluster groups and configure these groups to have different preferred owners. The postprocessor repository and the database then charge to different servers, and the servers act as failover for each other.

The postprocessor repository is included in the StreamServer setup. At installation, a Windows service for the postprocessor repository is set up automatically. Since the service will be started by the cluster, you must configure it to start manually.

You must set up a cluster service for the postprocessor repository. The cluster service will save the jobs on the clustered disk that resides in the same cluster group. Access to this disk runs through the cluster service. When LiveCycle Production Print ES is saving anything to the postprocessor repository, it is done through the cluster to which all StreamServer applications have access.

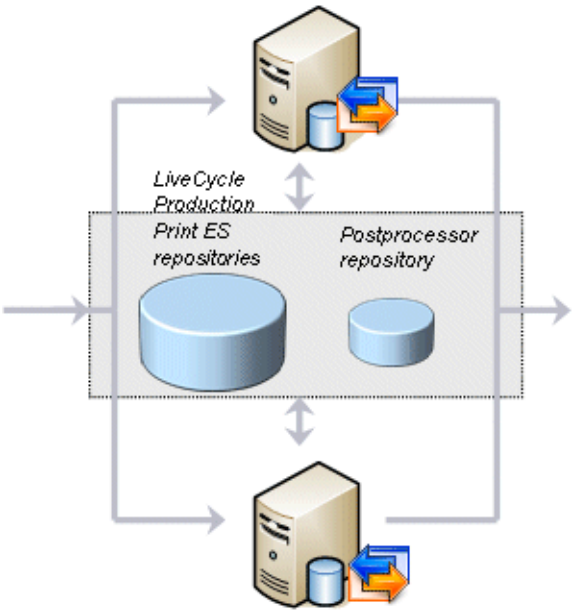

Figure 6. Example of a cluster with database and postprocessor repository

Setting up a postprocessor repository in a cluster

This section includes a few tips and recommendations for setting up the postprocessor repository in a cluster.

- Install the postprocessor repository locally on each node. The postprocessor repository is included in StreamServer setup. See the *Installation Guide*.
- Configure the Windows service for the postprocessor repository to start manually.
- Configure the repository-related parts of the Project in Design Center. See the postprocessor repository documentation.

  When addressing the postprocessor repository in the postprocessor repository output connector settings, it is recommended that you use Uniform Naming Convention (UNC) paths with the following syntax:

  ```
  \\<Cluster name>\<File share name>
  ```

- Specify the default database path. See "Specifying the default database path."
- Set up the required cluster resources in the cluster administration module:
  - A physical disk (if the repository resides in a separate cluster group).
  - A cluster service (for example, in Windows, a generic service). It is sufficient to select only cluster name as resource dependency since the cluster name itself is dependent on the IP address.

  For example, in MSCS:

LiveCycle Production Print ES Application Group	
Resource name	**Resource type**
Disk D:	Physical disk
Postprocessor repository	Generic service

Specifying the default database path

1. Open the `ptserver.cfg` file located in:
   `\StreamServe\Applications\StreamServer\<Version>\common\bin`
2. Make sure the DefaultDatabasePath points to the directory on the clustered disk residing in the same cluster group as the postprocessor repository, for example, `D:`.

   Note: Do not use UNC paths in `ptserver.cfg`.

Printers and print servers

If you intend to print the output from LiveCycle Production Print ES, the cluster must include one local print server for each node in the cluster and one virtual print server for the cluster. All available printers and port monitors from all local print servers must be installed on the virtual print server.

You must set up at least three cluster resources for the virtual print server: an IP address, a host name, and a print spooler. You can create a separate cluster group for the virtual print server, or you can include the virtual print server in an already existing group.

In UNIX, most often you use an Line Printer Daemon (LPD) server as print server.

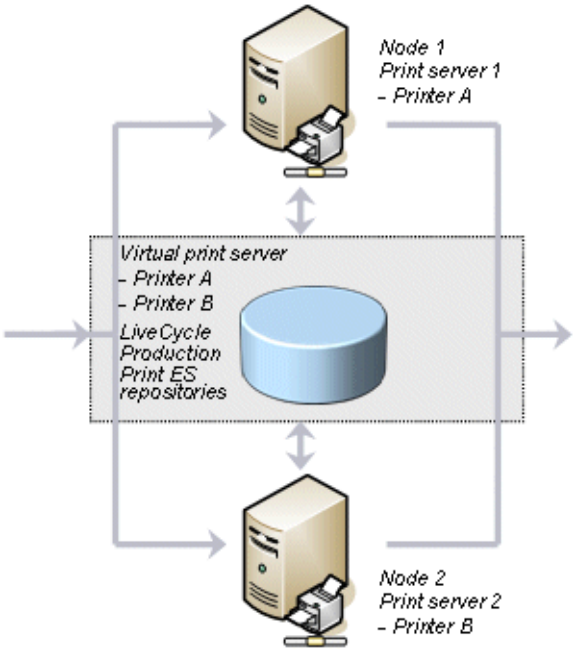

Figure 7. Example of printers and print servers in a cluster

Setting up printers and print servers in a cluster

This section includes a few tips and recommendations for setting up the printers and print servers in a cluster.

Windows

- Install a Microsoft virtual print server. For more information, see the Microsoft user documentation.
- If you want to reuse the printers and port monitors already installed on another print server on the virtual print server, you can use the Microsoft Print Migrator 3.1 tool (available for download from www.microsoft.com). For more information, see the Microsoft user documentation.
- If you intend to use LiveCycle Production Print ES EMF print processor and port monitor in a Microsoft cluster, you must install the print processor and port monitor on each node in the cluster. For more information, see the *Installation Guide*.

- Set up the required cluster resources in the cluster administration module:
  - A physical disk (if the printers reside in a separate cluster group)
  - An IP address
  - A network name
  - A print spooler

  For example, in MSCS:

Virtual Print Server Group	
Resource name	**Resource type**
Disk F:	Physical disk
Spooler resource	Print spooler
Spooler IP address	IP address
Spooler network name	Network name

UNIX

- Cluster the LPD servers using standard procedures.

StreamStudio

The StreamStudio web applications require the following services:

- A database instance (with LiveCycle Production Print ES repositories)
- A LiveCycle Production Print ES service gateway
- A Java™ application server
- A web server

To make the StreamStudio web applications highly available, you must make each one of these services highly available.

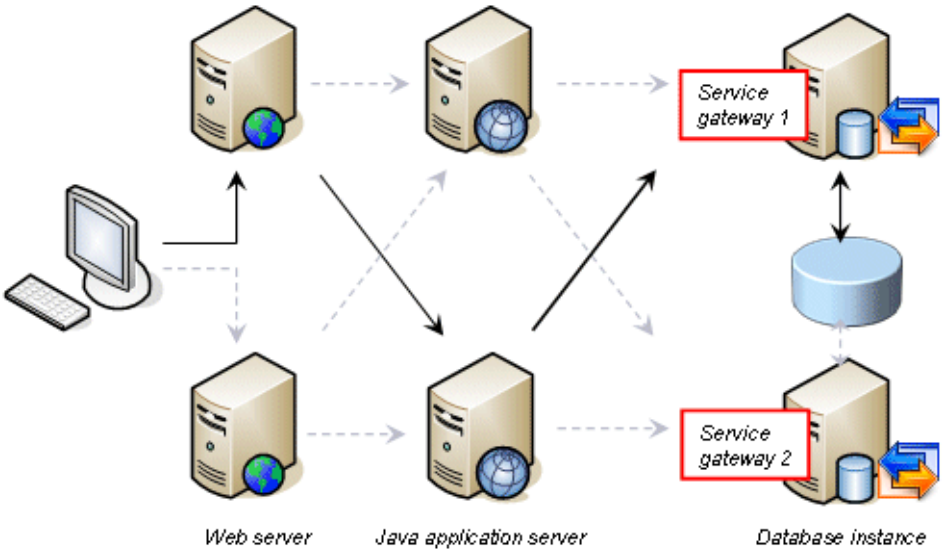

Figure 8. Example of a highly available StreamStudio environment

Making the database instance highly available

StreamStudio almost always shares the LiveCycle Production Print ES runtime repository, residing on the database instance (Microsoft SQL Server or Oracle), with other LiveCycle Production Print ES applications. The configurations, queues, and archived documents accessed via StreamStudio are stored in the database, which is cluster-aware.

You can achieve high availability for the database by configuring it in a failover cluster environment. For more information, see *Database for StreamServe repositories*.

Making the service gateway highly available

StreamStudio connects to StreamServer applications, databases, and user directories via a service gateway. The service gateway does not own jobs, data, queues, or documents, and it is not cluster-aware. However, you can make the service gateway highly available by using load balancing between two or more nonclustered nodes. StreamStudio then load balances the requests between multiple service gateway instances defined for the application domain in Control Center.

When creating the application domain, you can specify a primary and secondary service gateway instance for each Java application server. It is recommended that you let the Java application servers point to different primary service gateway instances. For example, application server 1 uses service gateway 1 as its primary instance, and application server 2 uses service gateway 2 as its primary instance.

Here are a few tips and recommendations for making the service gateway highly available.

- Install the service gateway instances on the same physical machines as StreamServer, or on the same physical machines as the Java application servers. The service gateway is included in the LiveCycle Production Print ES Framework and Control Center setup. See the *Installation Guide*.
- In Control Center, update the application domain:
  - For each host within the application domain, create a new application of type `Service Gateway <Version>`.
  - Open the Application Domain Editor and specify the new service gateway instances (primary and secondary).
  - Since the service gateway applications are not started by the cluster, set up the applications to start automatically.

For further details, see the Control Center documentation and the *StreamStudio Administrator´s Guide*.

Making the Java application server highly available

StreamStudio web applications run on a Java application server. They applications maintain user sessions, but do not own jobs, data, queues, or documents. The StreamStudio applications are not cluster-aware. However, you can make the Java application server highly available, either with failover clustering using third-party tools, or with load balancing between two or more nonclustered nodes.

The Java application server is a third-party product. Descriptions of how to configure third-party products is beyond the scope of this document. For information, see the user documentation for the Java application server.

If you use a default StreamStudio installation, read the following section for some tips.

Tips for using a default StreamStudio installation

In a default StreamStudio installation, using Apache Tomcat 5 as the Java application server and Apache HTTP Server 2 as the web server, load balancing between multiple Tomcat instances can be achieved by a JK 1.2 connector running on the web server. The JK connector directs user requests from the front-end web server to the back-end Java application server.

If there are multiple back-end Java servers, you can configure the JK connector to load balance requests between the Java servers. Configure the JK connector by updating a workers.properties file. This file lists so called workers (`w1`, `w2`, and so on). Each worker refers to a Java application server instance.

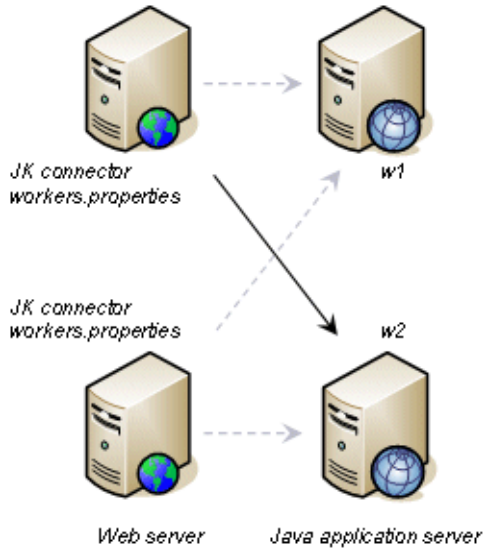

Figure 9. Example of highly available Java application servers

- Install the StreamStudio software and the Tomcat 5 Java application server on the different physical machines. See the *Installation Guide*.
- On each web server, configure the `mod_jk.conf` file and the `workers.properties` file to point to the possible back-end servers. The files are usually located in:
  - Windows: `C:\Program Files\Apache Group\Apache 2\conf`
  - UNIX: `/etc/httpd`

Following are file examples.

Example 1: mod_fk.conf

The `mod_jk.conf` file might look like the following:

```
# This is a sample mod_jk.conf file (referenced by httpd.conf) used
for

# JK 1.2 and Apache 2.x to connect to back-end AJP13 workers

# (which are defined in the workers.properties file)

LoadModule jk_module modules/mod_jk.so

JkWorkersFile conf/workers.properties

JkLogFile logs/mod_jk.log

JkLogLevel warn

JkLogStampFormat “[%a %b %d %H:%M:%S %Y] “

JkRequestLogFormat “%w %V %T”

JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories

# This directive tells the web server which URLs to assign to AJP
workers

JkMount /applications/* ajplb
```

Example 2: workers.properties

The `workers.properties` file might look like the following:

```
# This is a sample workers.properties file.

# This file is used by front-end web server(s) to enumerate

# connections to back-end Tomcat instance(s), using JK 1.2.

#

# For more information, refer to http://tomcat.apache.org/connec-
tors-doc

#

#

# Only the worker(s) listed here should be referenced by the httpd.
conf

# (or mod_jk.conf) or IIS registry file. If a load-balancing worker
is

# referenced here, do not duplicate its members in this list.

#

worker.list=ajplb

worker.maintain=60

#

# Define a load-balancing worker named “ajplb” with one member “w1”

# The balance_workers value is a comma-separated list, for exam-
ple:# “w1,w2,w3,w4”

#

#worker.ajplb.type=lb

worker.ajplb.balance_workers=w1,w2

worker.ajplb.sticky_session=True

worker.ajplb.sticky_session_force=False

#worker.ajplb.method=Request

#worker.ajplb.lock=Optimistic

#

# Define an ajp13 worker named “w1”

# Verify the host and port values, especially if the Tomcat
instance(s)

# are located on other server(s).

#

worker.w1.type=ajp13

worker.w1.host=node1.localdomain.com

worker.w1.port=8009

#worker.w1.socket_timeout=15#worker.w1.socket_keepalive=False
```

```
#worker.w1.retries=3

#worker.w1.connection_pool_size=1

#worker.w1.connection_pool_minsize=1

#worker.w1.connection_pool_timeout=0worker.w1.lbfactor=1

#

# Define an ajp13 worker named “w2”

#worker.w2.type=ajp13

worker.w2.host=node2.localdomain.com

worker.w2.port=8009

#worker.w2.socket_timeout=15

#worker.w2.socket_keepalive=False

#worker.w2.retries=3

#worker.w2.connection_pool_size=1

#worker.w2.connection_pool_minsize=1

#worker.w2.connection_pool_timeout=0

worker.w2.lbfactor=1
```

Making the web server highly available

StreamStudio uses a web server as a front-end proxy to a Java application server, on which the StreamStudio web applications actually run. The web server does not own jobs, data, queues, or documents. It can be made highly available with third-party tools, by using either failover clustering or balancing.

The web server is a third-party product. Descriptions of how to configure third-party products are beyond the scope of this document. For information, see the user documentation for the web server.

Testing the cluster

You can test the failover and load balancing of the LiveCycle Production Print ES cluster solution.

To test failover: Test 1

Open the cluster administration module and move the cluster groups manually while jobs are added for processing.

Make sure that:

- The cluster resources can be moved.
- The cluster resources are correctly started.
- LiveCycle Production Print ES continues to process the jobs.

To test failover: Test 2

While processing a job, kill one of the StreamServers.

Make sure that as soon as the second node has completed the currently processed job, it continues with the noncompleted job from the failed StreamServer.

To test load balancing

Add some larger jobs for processing and test load balancing between the nodes. Prerequisites:

- The scheduled polling interval (schedule spooling) is less than half the time one job takes to be processed.
- There are more jobs than number of threads for each StreamServer.

Make sure that the primary node processes the incoming jobs under low workloads. If workload exceeds the processing limit for the first node, the standby node should continue with the next job in the queue. One of the two nodes finds the first job, and the other node instantly finds the next job.

Adobe Systems Incorporated
345 Park Avenue
San Jose, CA 95110-2704
USA
www.adobe.com

Adobe, the Adobe logo, and LiveCycle are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. IBM and AIX are trademarks of International Business Machines Corporation in the United States, other countries, or both. Linux is the registered trademark of Linus Torvalds in the U.S. and other countries. Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. StreamServe is a trademark of StreamServe, Inc. Java and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. UNIX is a registered trademark of The Open Group in the U.S. and other countries. All other trademarks are the property of their respective owners.

© 2008 Adobe Systems Incorporated. All rights reserved. Printed in the USA.

95011727 9/08